React JS

What is react?

* It’s a library not a framework
* Focus only on UI, don’t focus on http calls/routings, only responsible for rich Ui
* Has rich ecosystem for other purposes.
* Its component base architecture.
* Can write reusable code, that component can be used in angular or vue.

Declarative Paradigm: just tell what to do, how it does is up to them.

Imperative paradigm: we explicitly say what to done step to step, we control the flow.

* React is declarative (tell what you want it build actual UI). Declarative paradigm in contrast to imperative paradigm we use to which implements algorithm explicit steps.
* We have to what to create react will create actual UI.
* It handles efficiently and updates and renders the exact component and updates the DOM.
* It can be integrated to any app as a portion or complete page or whole application.
* React native is used for mobile apps.
* Html, CSS, JavaScript, ES6,
* In JavaScript – ‘this’ keyword, filter, map and reduce.
* In ES6 – let & const, arrow functions, template literals, default parameters, object literals, rest and spread operators and DE-structuring assignment.

Folder structure

* We want react to control the app for that we have one div tag with id = root one time react app will take control and it responsible rendering. (this is called as root DOM node)
* This root is tagged to index.js file with that elementById(‘root’).

Component structure:

* Components can contain many components ex: app component (acts as parent component)
* In react we have two types of components:

1. Stateless Functional component
2. Stateful class component

|  |  |
| --- | --- |
| Stateless functional component | Stateful class component |
| 1. Are literally JS functions they return html which describe the UI. | Class extending component calss.  Render method returning HTML |

Functional vs Class Components

|  |  |
| --- | --- |
| Functional component | Class component |
| 1. Simple functions receiving props and returning declaration. 2. Use functional component as much as possible. 3. Absence of ‘this’ keyword. 4. Solution without using state 5. Mainly responsible for the UI 6. Also called as Stateless/Dumb/Presentational | 1. More feature rich 2. Maintain their own private data – state 3. Complex UI logic 4. Provide lifecycle hooks 5. Stateful/Smart/Container components |

JSX (JavaScript XML)

Writ XML-like code for elements and components.

JSX tags have a tag name, attributes and children.

JSX makes code simpler and elegant. (Not necessity to use JSX)

JSX ultimately transpiles to pure JS which is understood by the browser.

**JSX differences**

Class -> className

For -> htmlFor

CamelCase property

* onclick -> onClick
* tabindex -> tabIndex

**Props**: are immutable

React components should work as pure function with respect to props parameter cannot change at any time.

All the application make use of props.

**Props VS State**

|  |  |
| --- | --- |
| Functional component | Class component |
| 1. props get passed to the components 2. Functional parameters 3. props are immutable 4. Accessed as props – Functional Components 5. Accessed as this.props – Class Component | 1. State is managed within the component 2. Variables declared with in the function body 3. It is managed within the component so it has full control 4. Accessed as useState Hook – Functional Components 5. Accessed as this.state – Class Components |

**Do’s and DoNot’s**

* We should never modify **state** directly. Use **setState**.
* Whenever you want to execute code after state has changed don’t place code after **setState** method instead place that in callback function, and pass it as a second parameter for **setState** method.
* If you want to update state base on the previous state make sure pass function as an argument instead for regular object.
* Only place we can assign state is constructor, any other place if you want to update state you have to use **setState** method.
* Call to **setState** are asynchronies, if you place next to setState it will not update.
* React group multiple **setState** call to single update for better performance.

**Destructuring Props and State:**

* Destructuring in the parameter.
* Destructuring in function body.
* You can extract necessary props form the props object instead of all.

**Event Handling**

* Ex: onClick={clickHandler}
* We should not use as a function call Ex: onClick={clickHandler()} this will get trigged when the component get rendered. This comes worst in class component.
* In class component Ex: onClick={this.clickHandler}

**Bind Events**

1. Ex: onClick={this.clickHandler.bind(this)} this is the default functionality of JS by using bind(this).
2. The approach by using arrow function Ex: onClick={() => this.clickHandler()}
3. Binding in the constructor Ex: this.clickHandler = this.clickHandler.bind(this)